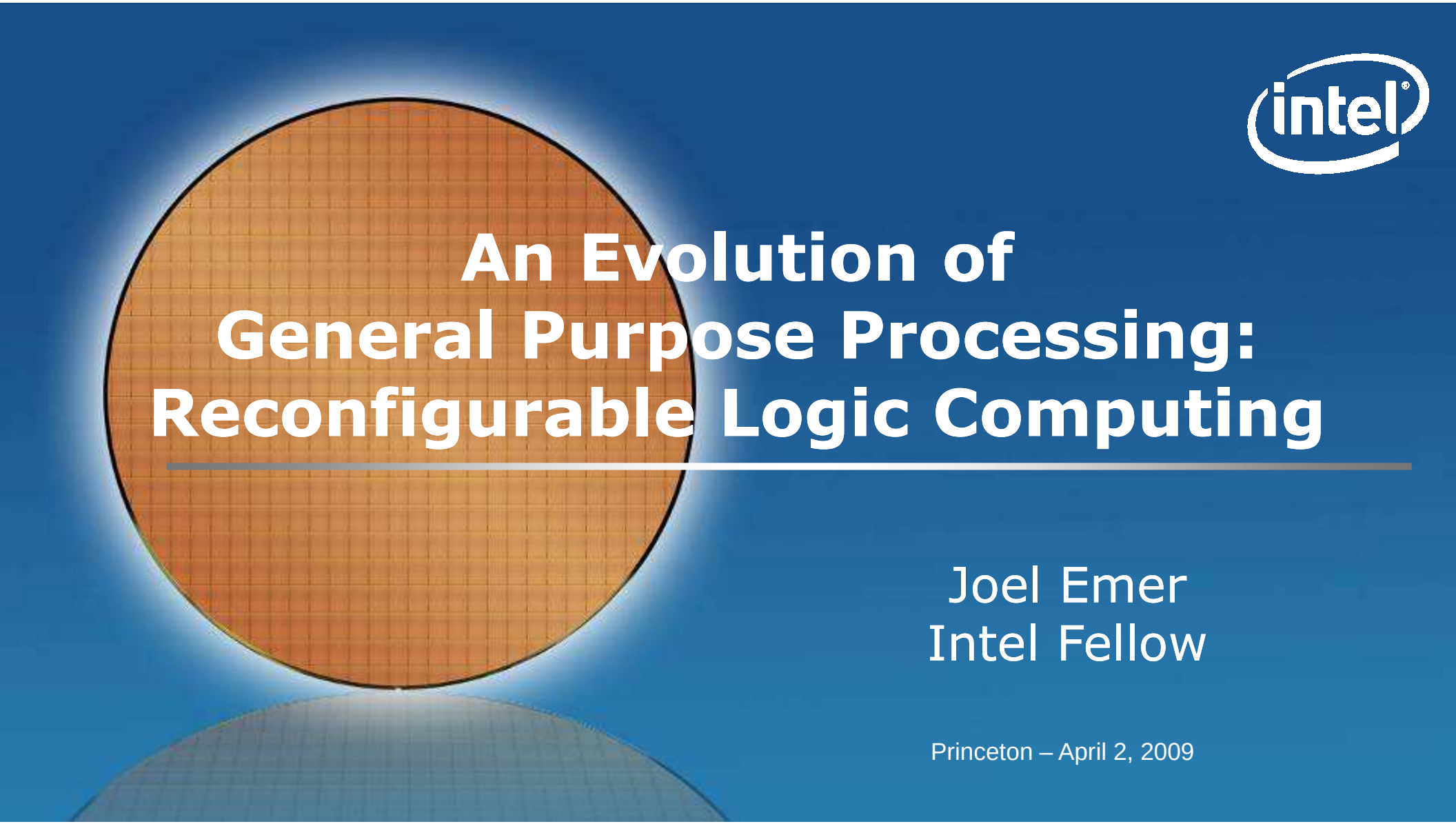


An Evolution of General Purpose Processing: Reconfigurable Logic Computing

Joel Emer
Intel Fellow

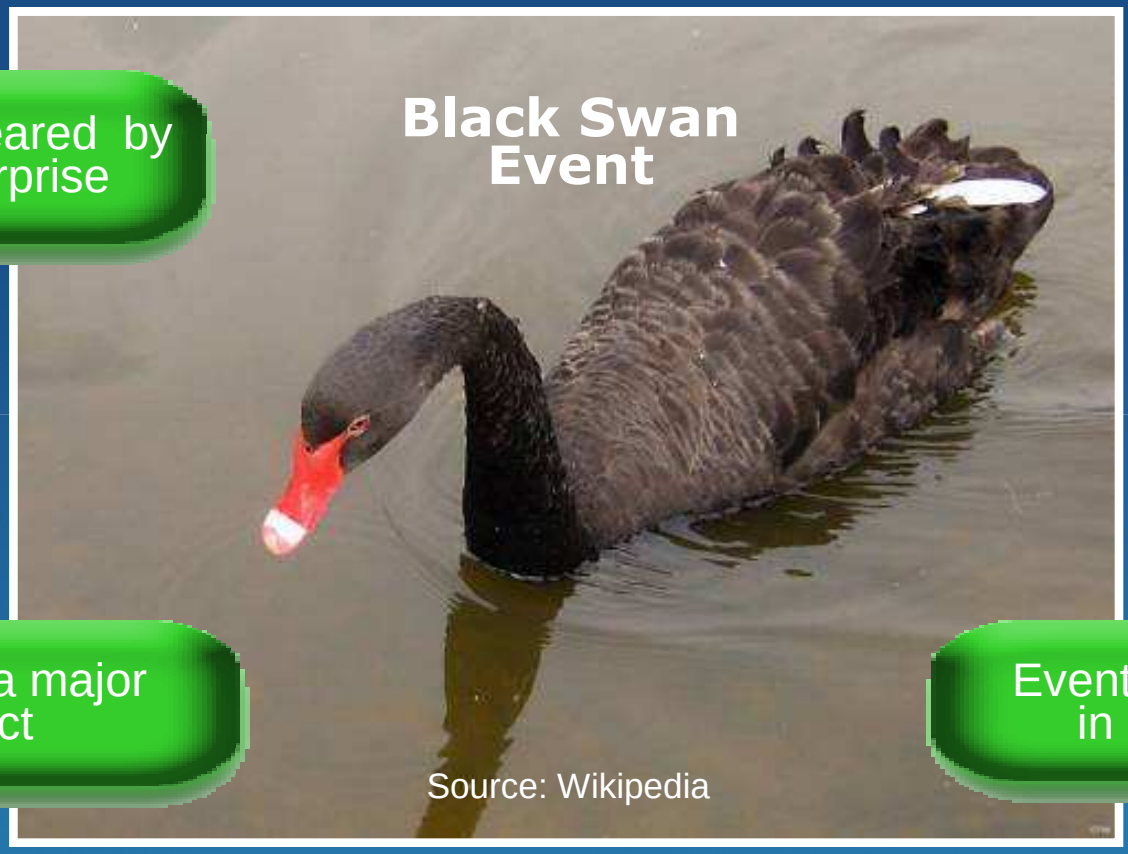
Princeton – April 2, 2009

Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.
- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- This document may contain information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.
- Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
- Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.
- Wireless connectivity and some features may require you to purchase additional software, services or external hardware.
- Nehalem, Penryn, Westmere, Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user
- Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.
- Intel, Intel Inside, Pentium, Xeon, Core and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
-

The "Black Swan" Theory

Event has appeared by complete surprise



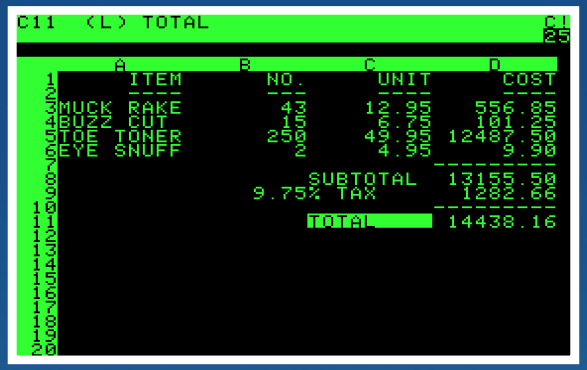
Source: Wikipedia

Event has a major impact

Event is explicable in retrospect

Note: The theory was described by Nassim Nicholas Taleb in his 2007 book *The Black Swan*

Black Swans in Computing



First Spreadsheet, 1983
VisiCalc Advanced Version

Desktop Publishing, 1985

Source: Wikipedia

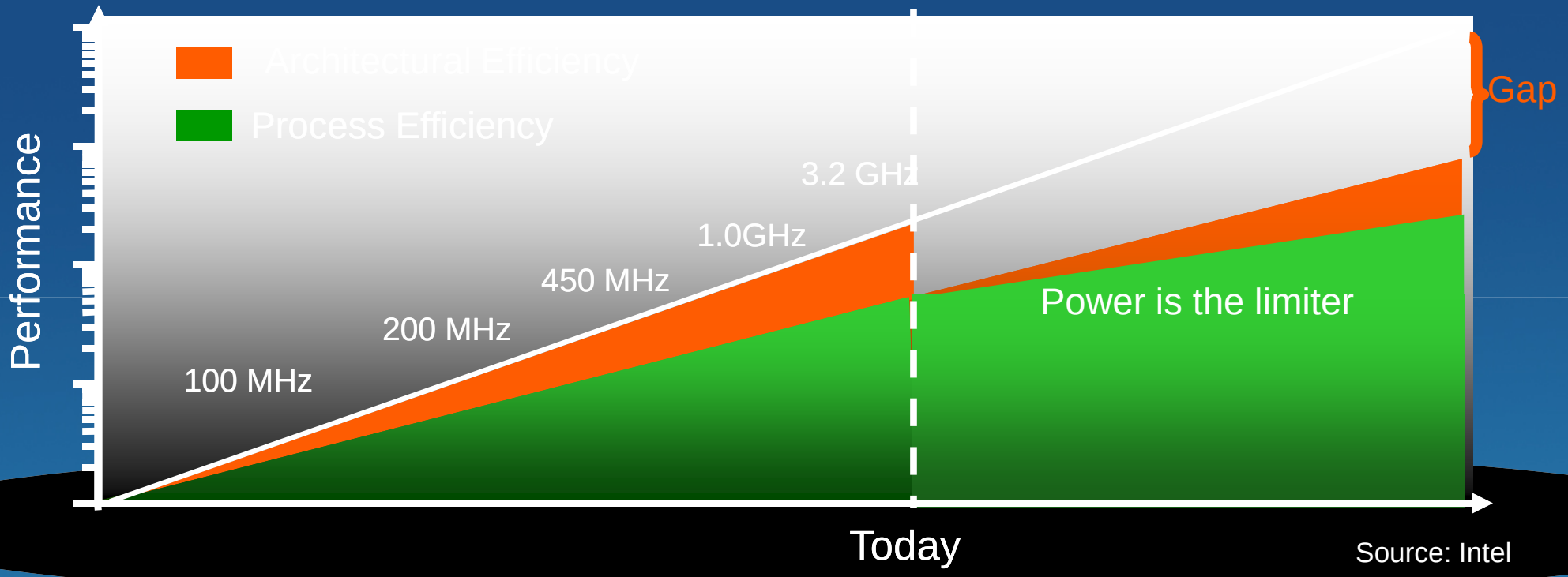
WIMP, 1980
Window, Icon, Menu, Pointing device

Computer Graphics

Another Black Swan

The Rise of the Internet

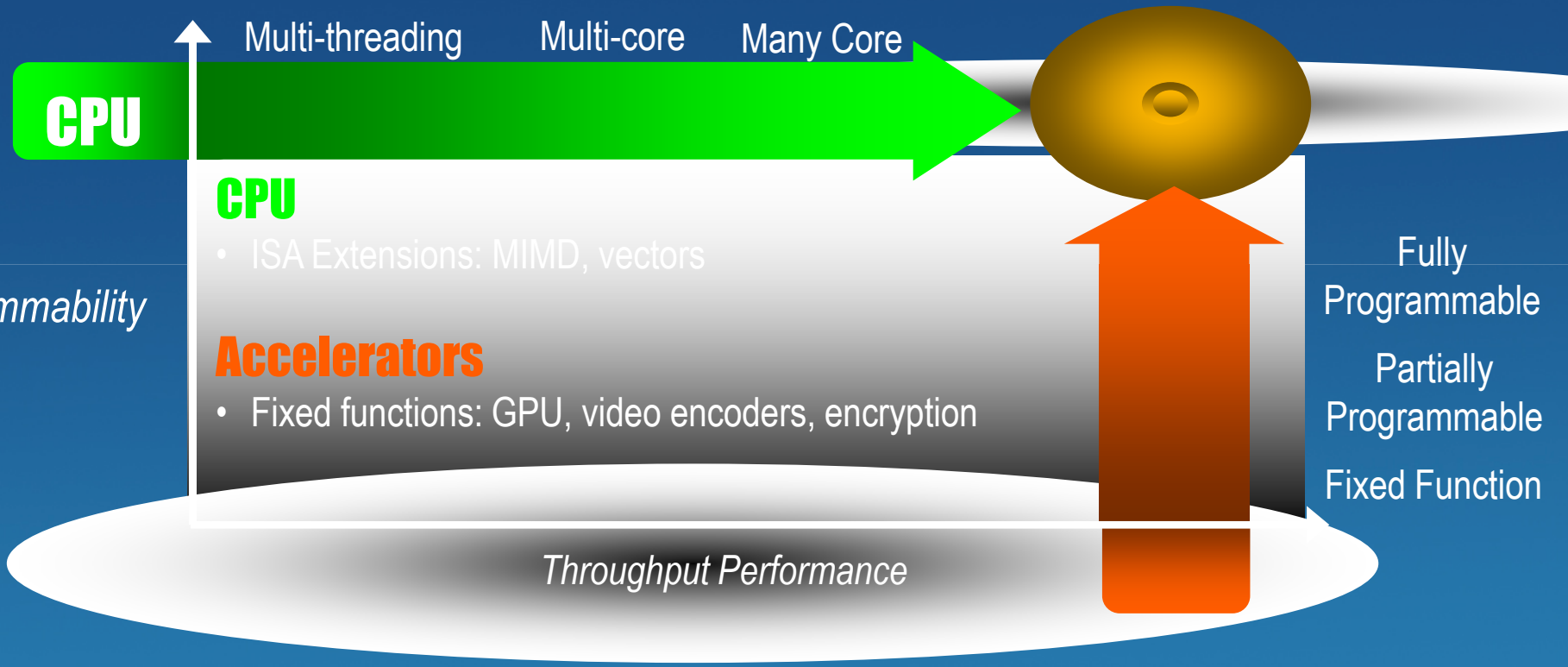
Breeding Black Swans



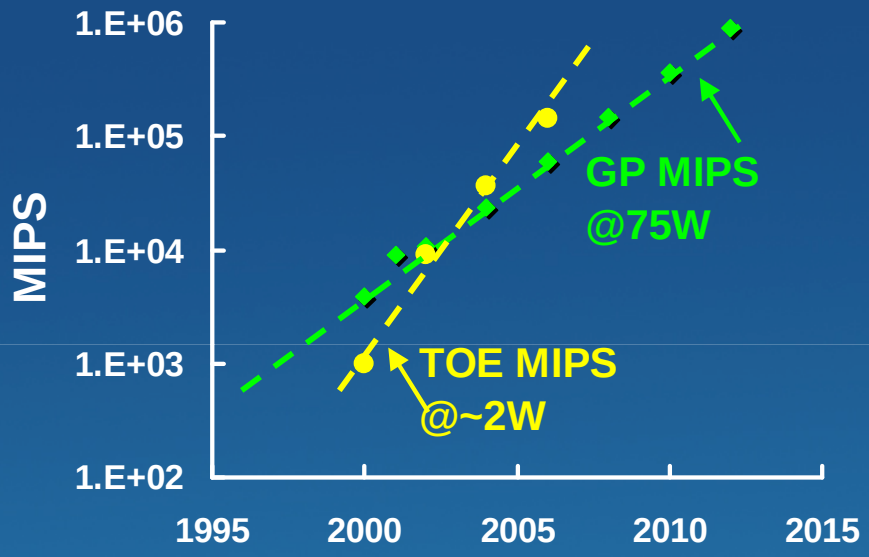
Demand architectural innovations to fill in the gap

Architecture Evolution:

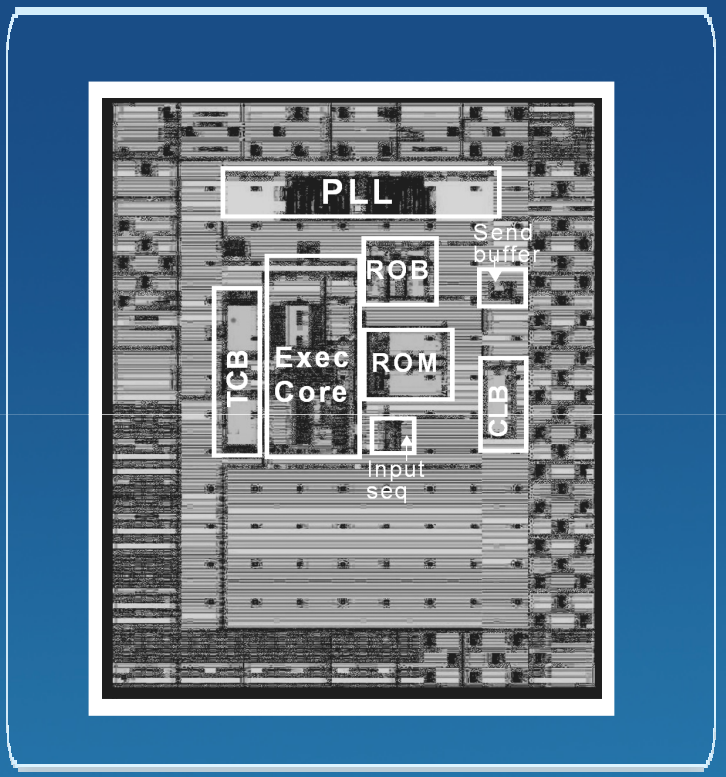
Addressing the Gap



Today's White Swan in Computing



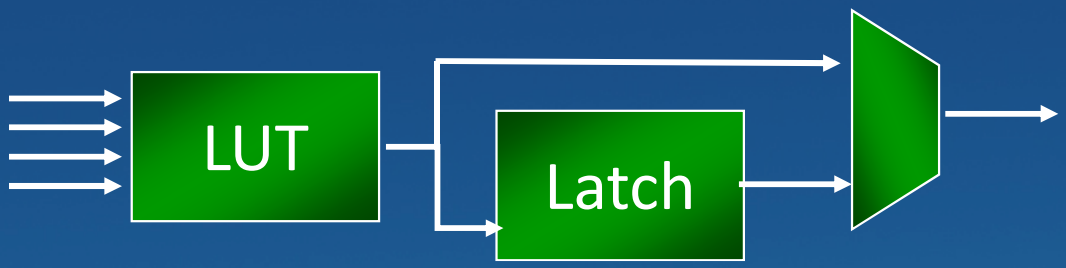
TCP Offload Engine
260K Transistors



Source: Intel Labs

Special Purpose Accelerators

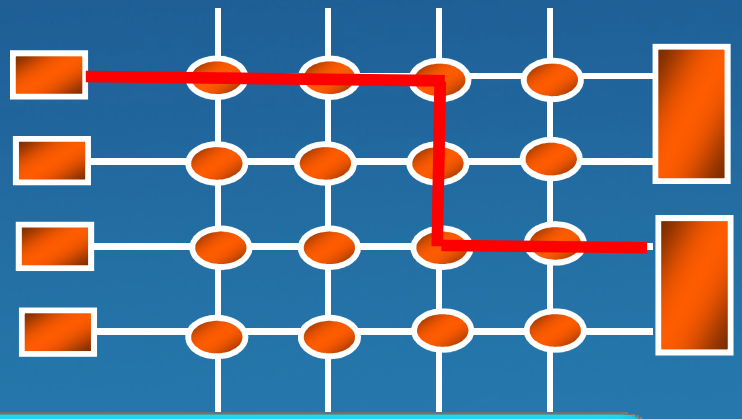
Field Programmable Gate Arrays (FPGA)



And	
00	0
01	0
10	0
11	1

Or	
00	0
01	0
10	1
11	1

.....



A fixed function compromise

Evolving reconfigurable logic usage

Logic replacement

- Low design cost and effort
- For low volume applications
- Often replaced with ASIC as volume increases

Algorithmic Computation

- Offloads a general purpose processor
- Used for many algorithms
- ASIC replacement not expected

Performance Acceleration with FPGA-based accelerators*

Applications	HW (FPGA)	SW Only
Hough & inverse Hough processing	2 seconds of processing time @20Mhz **370x faster**	12 minutes processing time Pentium 4 - 3Ghz
AES 1MB data processing/cryptography rate Encryption Decryption	424 ms/19.7 MB/s 424 ms/19.7 MB/s **13x faster**	5,558 ms / 1.51 MB/s 5,562 ms / 1.51 MB/s
Smith-Waterman ssearch34 from FASTA	100 sec FPGA processing **64x faster**	6461 sec processing time Opteron
Multi-dimensional hypercube search	1.06 Sec FPGA@140Mhz Virtex II **113x faster**	119.5 Sec Opteron - 2.2 Ghz
Monte-Carlo Analysis 64,000 paths	10 sec of Processing @200 Mhz FPGA system **10x faster**	100 sec processing time Opteron - 2.4 Ghz
BJM Financial Analysis 5 million paths	242 sec of Processing @61 Mhz FPGA system **26x faster**	6300 sec processing time Pentium 4 – 1.5 Ghz
Black-Scholes	18 msec FPGA@110Mhz Virtex-4 **203x faster**	3.7 Sec 1M iterations Opteron - 2.2 Ghz

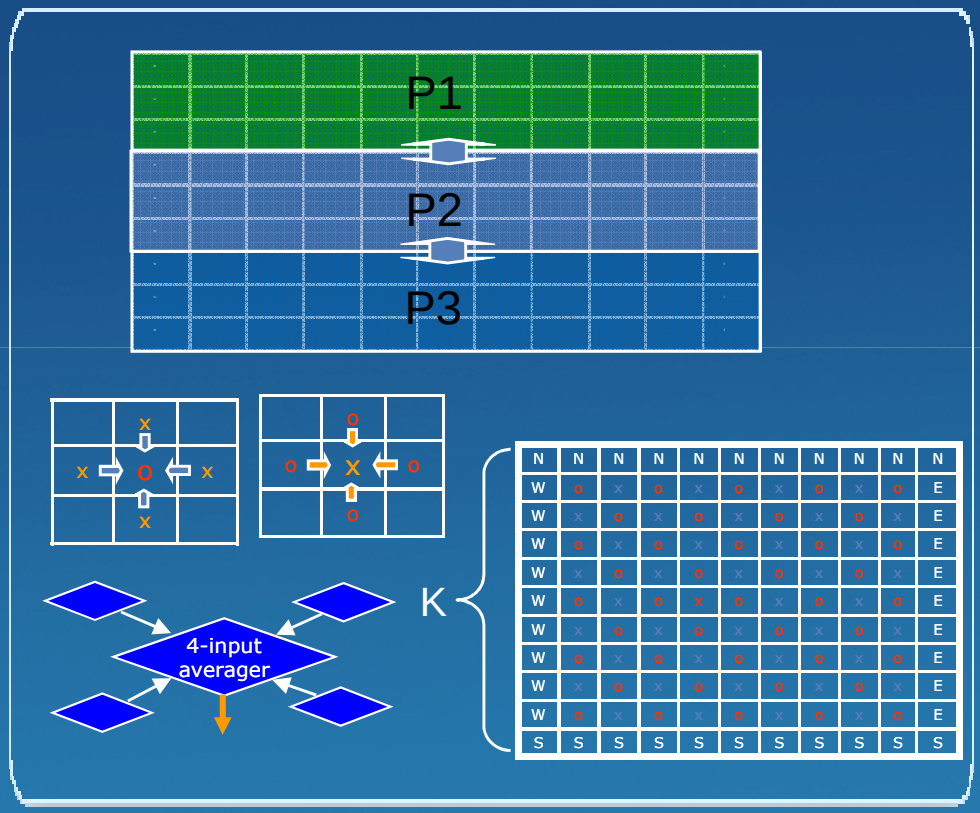
*** Chart from Celoxica**
HPC Accelerator Whitepaper Rev 0.9 (Intel), September 14, 2006, by Steve Duvall, Tom Marchok

Fine grain parallelism and state

- Example: numerically solving partial differential equation – Laplace's equation

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

- In traditional cores
  - Max parallel degree: *p* (threads)
  - K^2/p cycles to do one iteration
- In RL
  - Max parallel degree: $K^2/2$
  - 2 cycles to do computation in one iteration: one for all "o", one for all "x"
  - Note: the sequential version of this algorithm is not suitable in traditional cores with cache – break the law of spatial locality



Source: Intel Labs, Tao Wang

The Good

- Custom operations/data types – RL allows custom operations/data types
- Fine grain parallelism - replicated logic permits easy parallelism
- Local access to state - local state elements allows parallel state access
- Custom communication - explicit direct inter-module communication
- Flexible flow control - Control flow based on arbitrary state machine
- Better power efficiency - its custom computation / logic & interconnect

The Bad and the Ugly

Insufficient capacity

- Being overcome by Moore's Law
- Addressable in system architecture

Slower Cycle Time

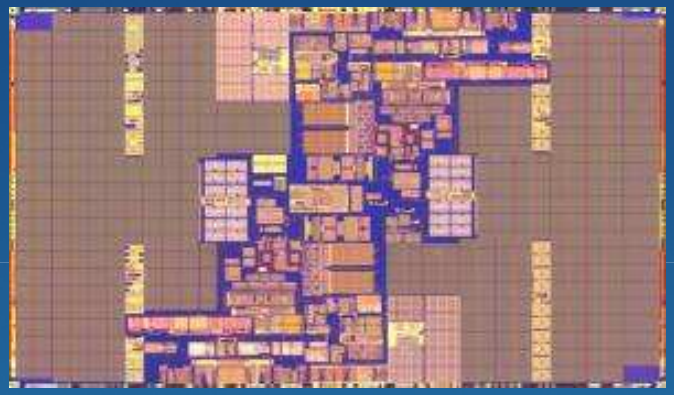
- Parallelism is already offsetting lower frequency
- Being addressed by higher-level/asynchronous fabrics

Difficult to Program and Debug

- Applications typically must develop everything
- No computation or system architecture
- No standard environment

Is there space for RL on-die?

Montecito



2 cores

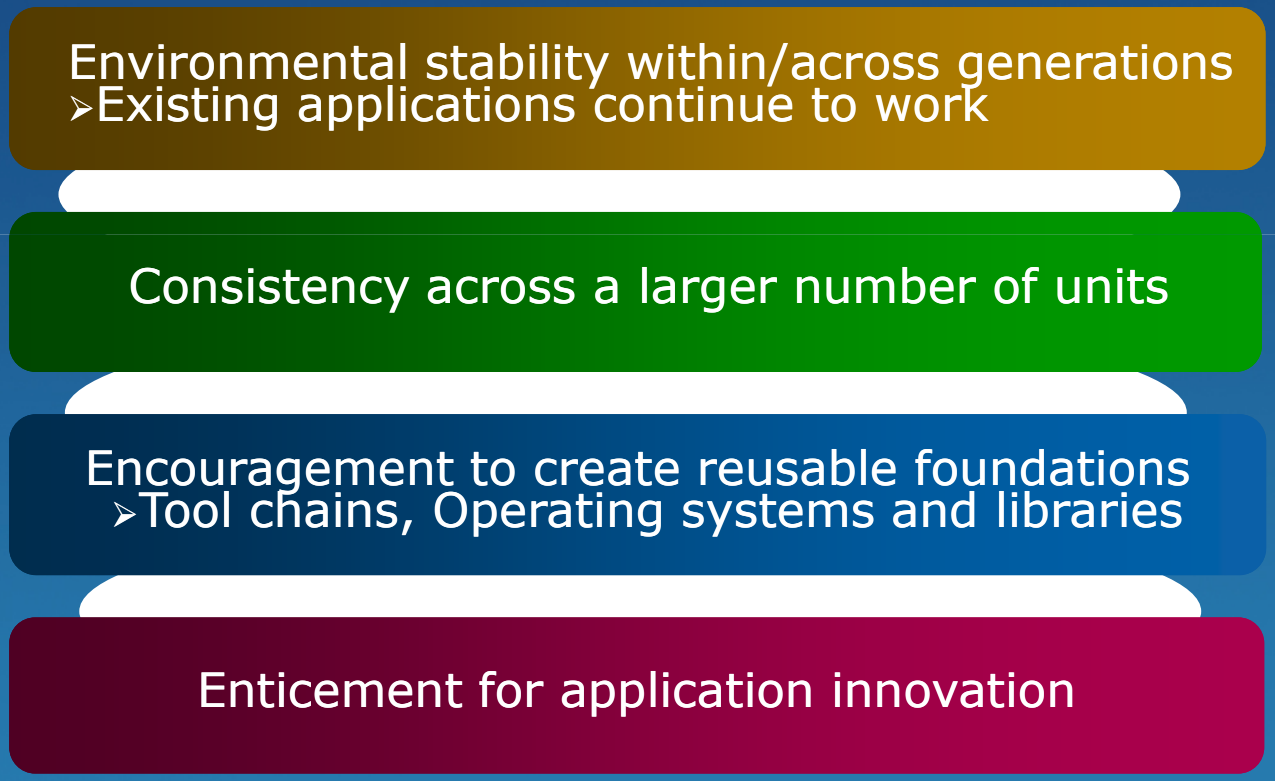
28.5M Transistors each

24MB L3 Cache

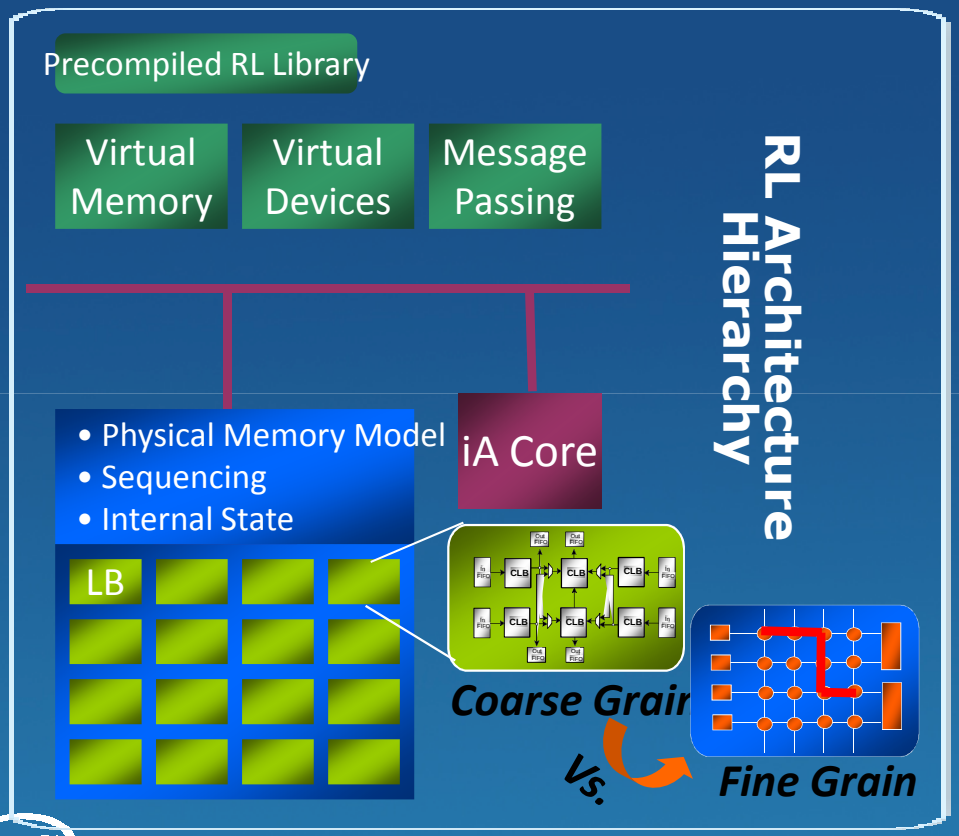
1550M Transistors

Architecture: Black Swan Enabler

Architectures provide:



RL architecture



raries

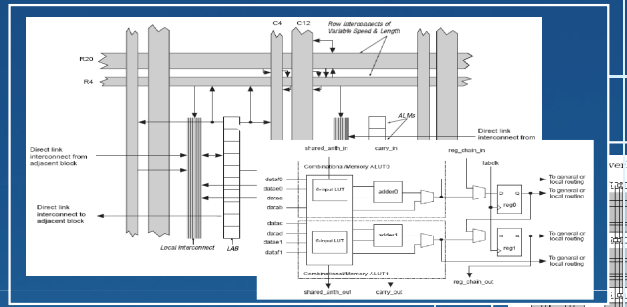
System Architecture

Compute Architecture

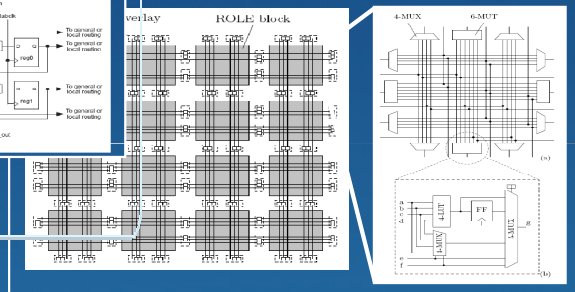
Fabric Architecture

An architectural Approach to Reconfigurable Logic

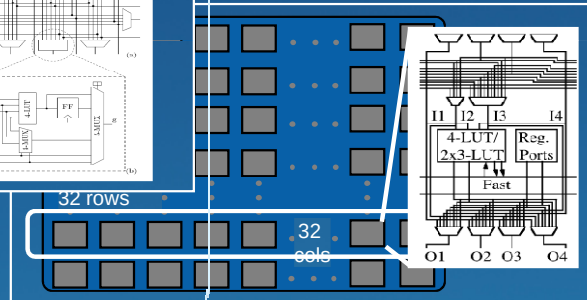
RL fabric architectures



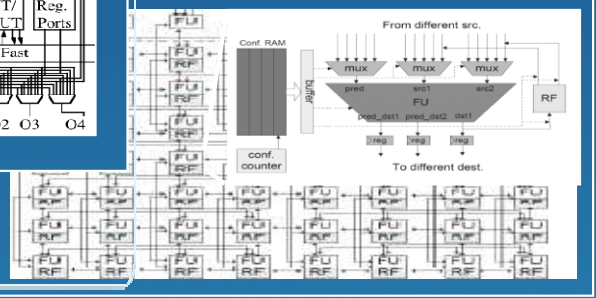
Fine-grain FPGA [1]



FPGA with flexible routing/logic block [2]



Row-based RL [3]
32 blocks/row for 32-bit ops



Coarse-grain RL [4]
Array of ALUs with fixed functions

Finer (flexibility)

Coarser (performance)

CPU and RL Compute Architectures

Small/Fast (registers)
Large/Slow (memory)

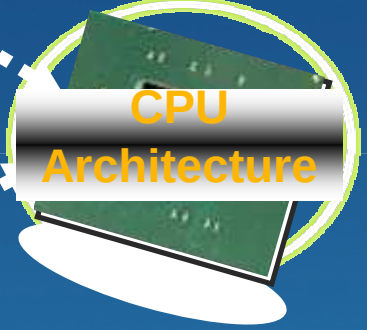
State

Does the RL have local state?
Can the RL access memory directly?

A fixed set of data types
and operations (ISA)

Operations
Data Types

Custom RL logic



Sets of sequential
and non-sequential
operations

Control
Flow

How are RL operations
sequenced?

A CPU architecture is a specification of the interface between the machine language and the hardware.

RL Compute Architecture Alternatives

Architectural semantics			Possible name of this kind of architecture	Example of what RL functions as
Async	Accesses memory	Has context		
0	0	0	Functional RFU	new bit manipulation instruction
0	0	1	Stateful RFU	accumulating data reduction instruction
0	1	0	memory-enabled RFU	memory-memory vector unit
0	1	1	memory-enabled stateful RFU	register-based vector unit including scatter-gather
1	0	0	functional accelerator	?
1	0	1	asynchronous RL accelerator	data-fed outboard accelerator
1	1	0	streaming RL accelerator	network adapter
1	1	1	peer RL processor	full function RL processor

RL Architecture Hierarchy

How do we sequence RL?

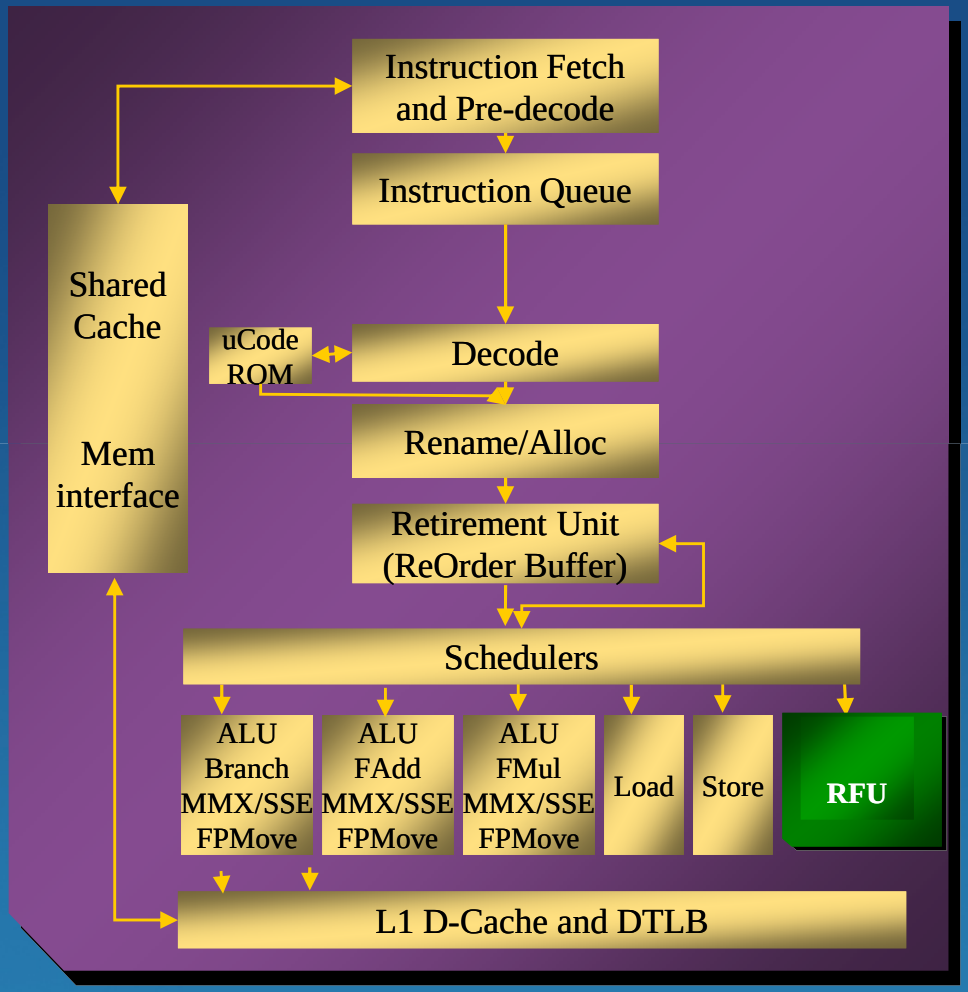
Synchronous

- Use RL operations inside a conventional pipeline. E.g., as a separate function unit
- Control handled by standard control instructions

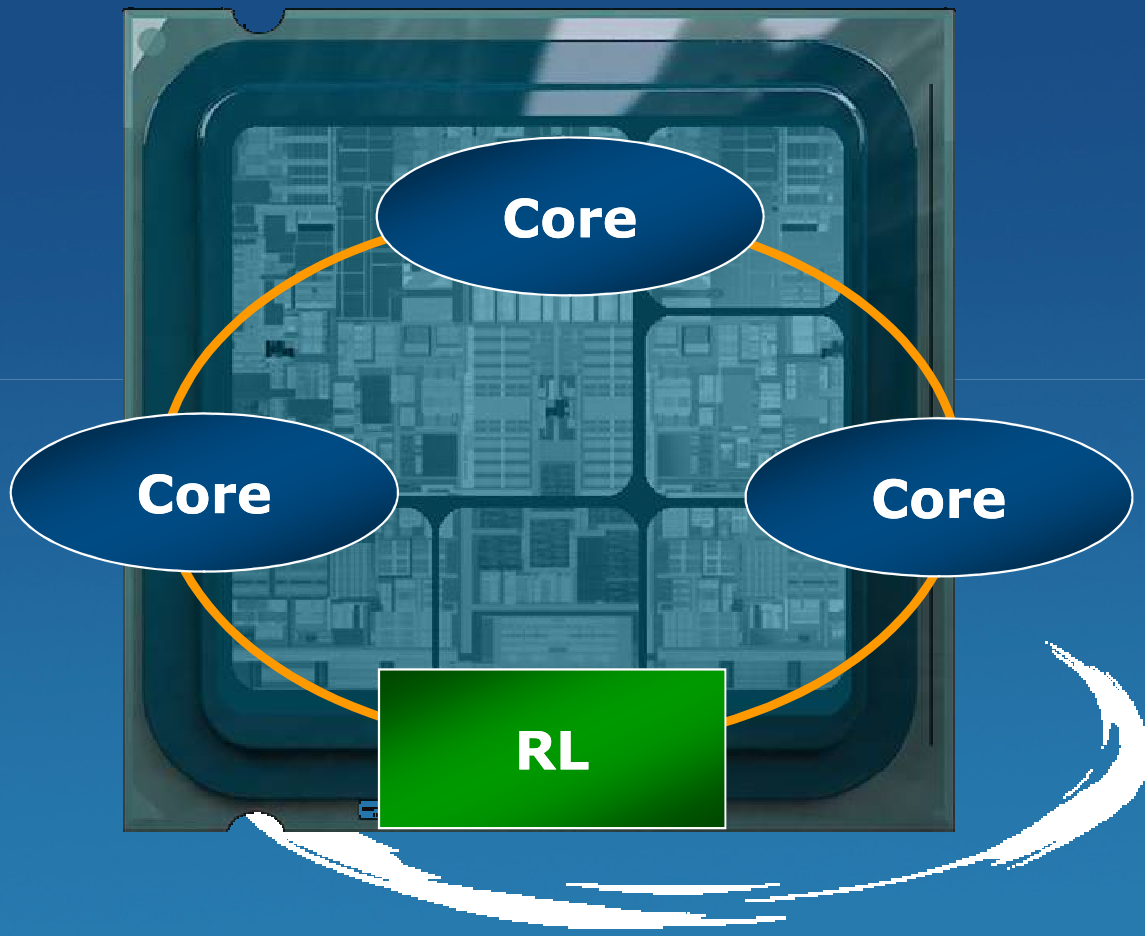
Asynchronous

- A standalone logical state machine
  - Implemented directly in RL
  - Allowing direct control input from any module

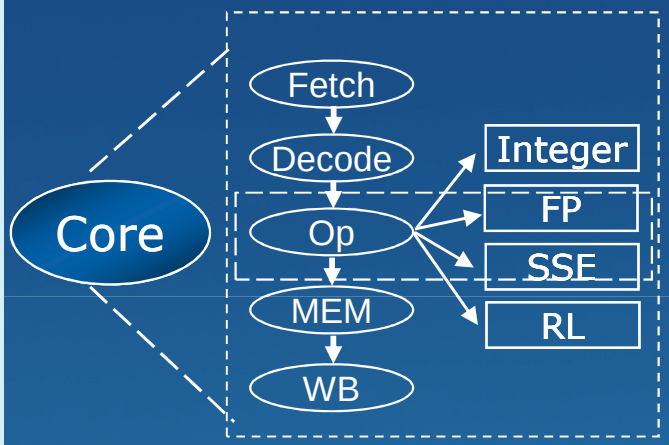
In-pipeline core-RL architecture (Type 0-3)



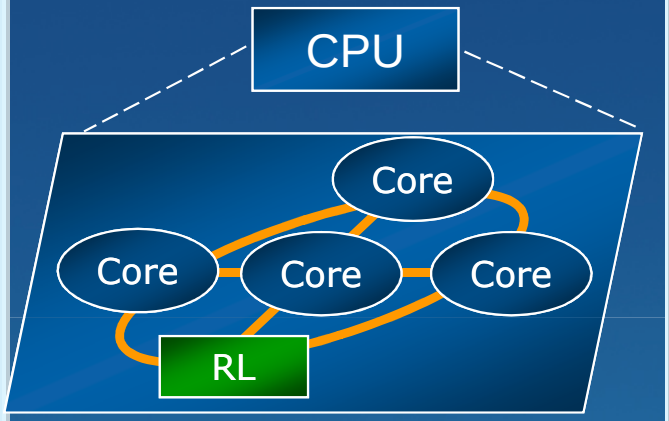
Peer computing RL architecture (Type 7)



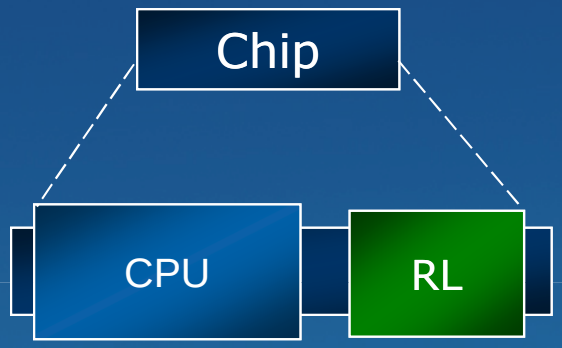
Implementation alternatives



Tightly coupled
Normally reconfigurable
instructions

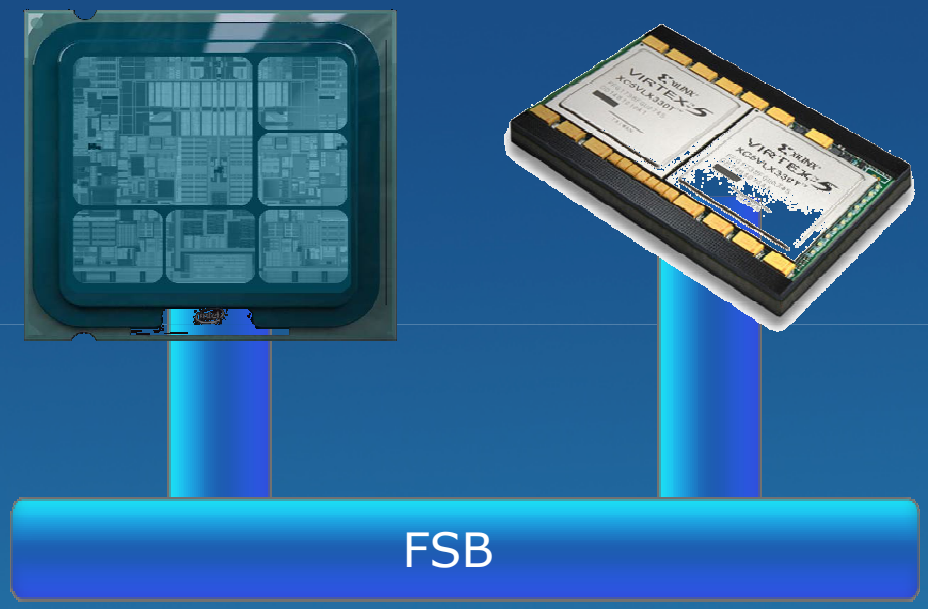


Medially coupled
Normally computing
blocks



Loosely coupled
Normally complex
application

A Processing Black Swan?



CPU and RL
Respective Strength

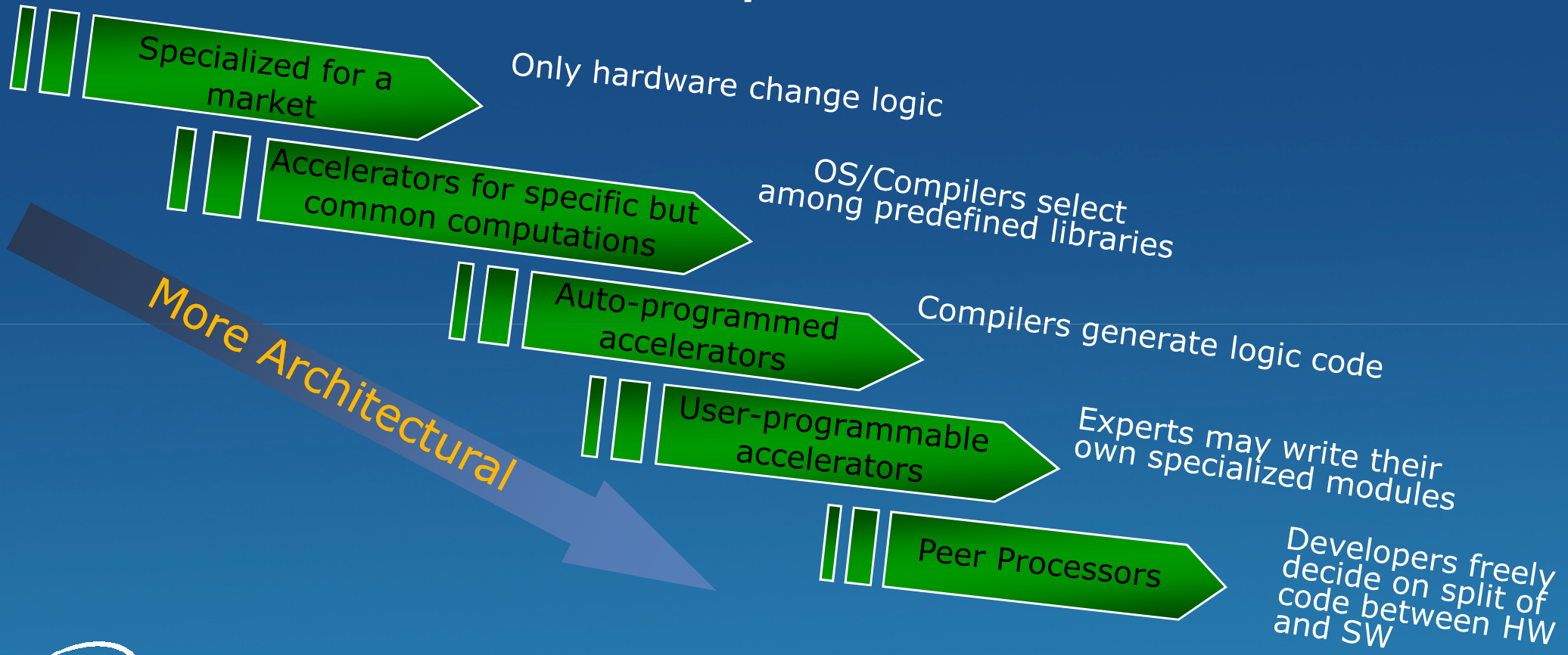
CPU

- Sequential, coarse parallel or un-pipelined algorithms
- Floating Point

RL

- Fine grained parallel or pipeline-parallel algorithms or complex flow control
- Custom operations, e.g. odd data sizes or fine grained bit-manipulations
- Integer

RL Development Model



The quality of the environment will determine the deployment model

System Environment Evolution

Software – Then

- Languages
  - Binary
  - Assembly
- No Standardized System Environment
  - Raw Devices
- No Distributed Computing Paradigms

Software – Now

- Languages
  - C++
  - Python
  - AJAX
- Rich System Environment
  - Device Abstractions
    - File Systems
    - Character Devices
  - Virtual Memory
  - Exception Handling
- Communication Paradigms
  - Shared Memory
  - Message Passing
  - Remote Procedure Calls

Evolving RL Systems

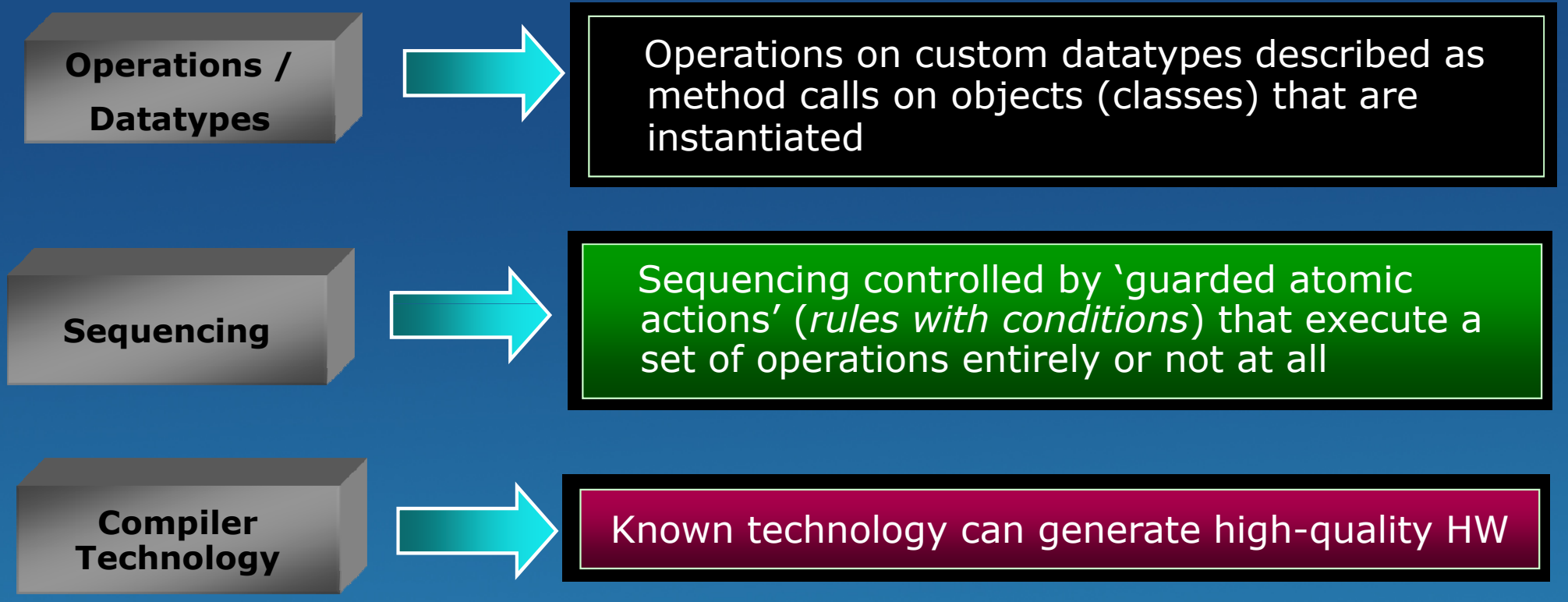
FPGAs – now

- Languages
  - Verilog (~Assembly)
  - VHDL (~Assembly)
- No Standardized System Architecture
  - Raw Devices
- No Distributed Computing Paradigms

FPGAs – looking forward

- Languages
  - C/C++
  - Bluespec
- Standardized System Architecture
  - FPGA virtual platform
- Communication Paradigm
  - Streams
  - Remote Request Response

Bluespec Model



Reed Solomon Results

WiMAX requirement is to support a throughput of 134Mbps

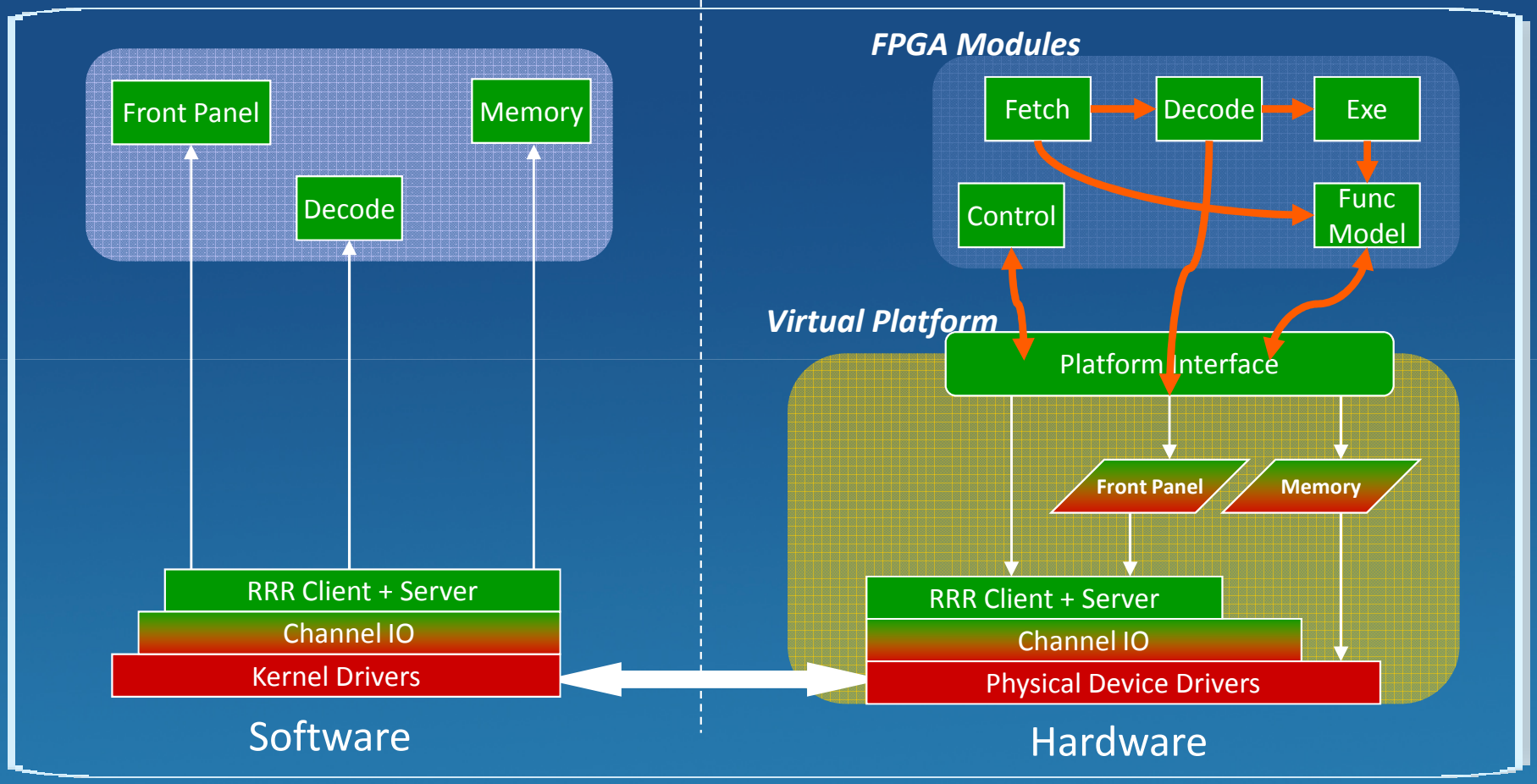
	Xilinx IP	Catapult-C	Bluespec
Equivalent Gate Count	297,409	596,730	267,741
Frequency (MHz)	145.3	91.2	108.5
Steady State (Cycles/Block)	660	2073	276
Data rate (Mbps)	392.8	89.7	701.3

Lower is better

Higher is better

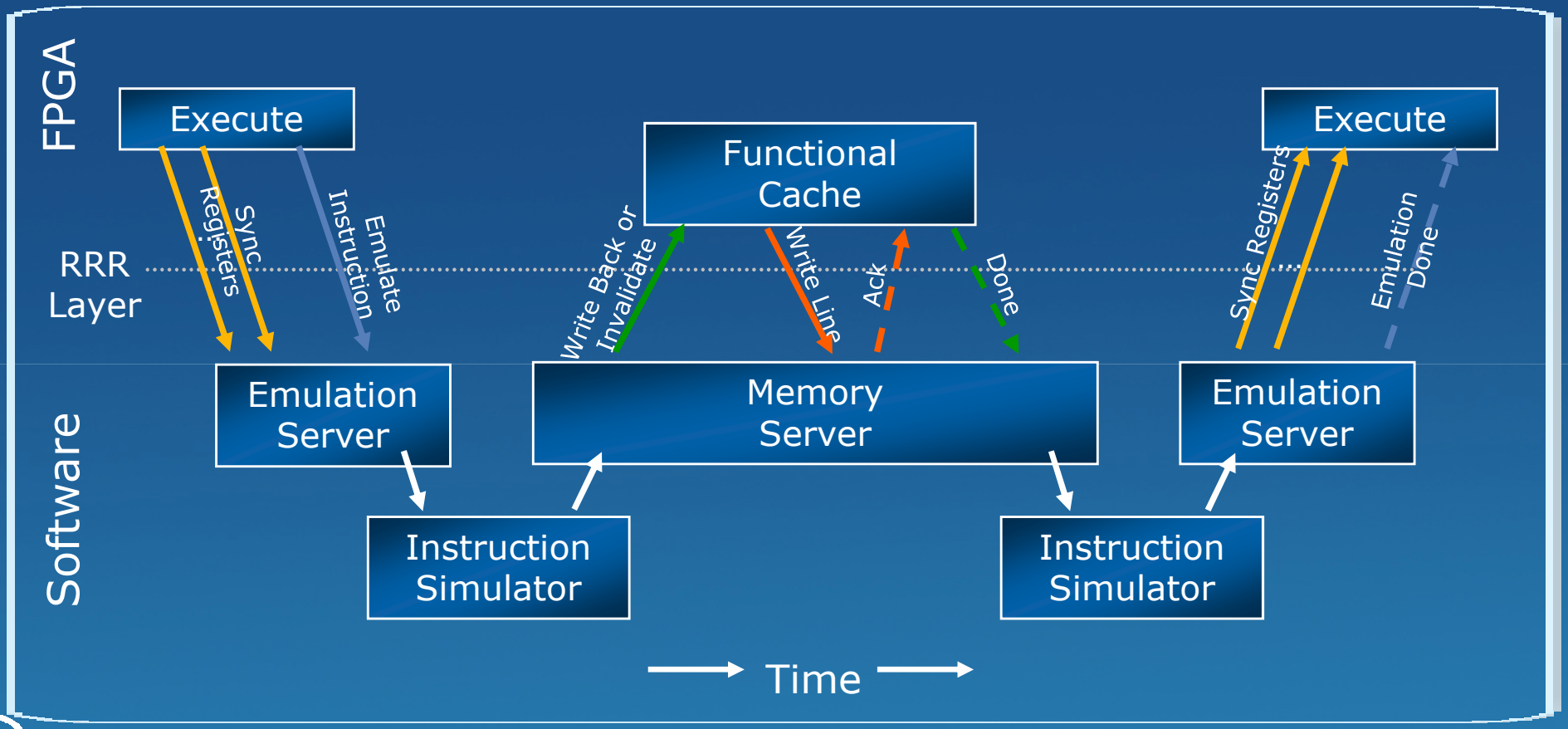
Source: MIT, Abhinav Agarwal, Alfred Ng – CSG

Virtual Platform



Source: Intel, Angshuman Parashar - VSSAD

Hybrid Instruction Emulation



Source: Intel, Michael Adler - VSSAD

Implemented in a day

Summary

A perspective on the role of general purpose computing in application innovation

Some possibilities for reconfigurable logic-based computing as a component of the general purpose computing environment

New opportunities for application of code generation and optimization

Acknowledgements

Arvind

Michael Adler

Azam Barkatullah

Angshuman Parashar

Michael Pellauer

Tao Wang

ZhiHong Yu

Questions?